Foreword

Vegter, Gert; Yap, Chee K.

Published in:
Mathematics in computer science

DOI:
10.1007/s11786-011-0088-z

Publication date:
2010

Math.Comput.Sci. (2010) 4:385–387
DOI 10.1007/s11786-011-0088-z

Mathematics in Computer Science

Foreword

Gert Vegter · Chee K. Yap

Published online: 18 October 2011

Theoretical algorithms and their complexity are central in foundational research at the interface between mathematics and computer science. There are many current challenges especially at the interface of continuous and discrete computation. For instance, how can we faithfully and efficiently discretize a continuous geometric object, or a continuous problem? Efficient new algorithmic techniques must be developed and analyzed. The inherent complexity of such problems needs to be characterized. Such considerations are central to current research areas such as (i) robust geometric algorithms, (ii) the surface–surface intersection (SSI) challenge in geometric modeling, (iii) mesh generation with topological guarantees, (iv) the development of a theory of real computation, and (v) the emerging field of numeric-algebraic computation. Such problems are the focus of this special issue.

In their survey paper on computational conformal geometry, D.X. Gu, F. Luo and S.-T. Yau focus on computational methodologies on discrete surfaces to discover conformal geometric invariants. Their paper addresses the construction of faithful discrete representations of smooth geometric structures on surfaces. The focus is on two major approaches, holomorphic differentials based on the theory of Riemann surfaces, and Ricci curvature flows on surfaces. The central issue is the construction of discrete versions of geometric entities like Gaussian curvature, Riemannian metrics, conformal structures, harmonic maps and Ricci flows. Well-known structures, like circle packings, play a key role in this context. As a proof of concept the authors present some applications of conformal geometric methods in several engineering fields, such as computer graphics (surface patch parametrization), computer vision (surface matching), geometric modeling (construction of manifold splines) and medical imaging (flattening of brain and colon surfaces).

During the last decade, meshing smooth hypersurfaces in Euclidean spaces has been a central theme in computational geometry. Methods for regular isolevel sets of smooth functions have been designed based on Delaunay tessellations of the ambient Euclidean space, and complexity bounds for optimal meshes with respect to Hausdorff distance between the implicit hypersurface and its approximating mesh have been derived. In their contribution to this special issue J.-D. Boissonnat and A. Ghosh address the even more challenging problem of certified meshing

G. Vegter (✉)
Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen,
Nijenborgh 9, 9747 AG Groningen, The Netherlands
e-mail: G.Vegter@rug.nl

C. K. Yap (✉)
Courant Institute of Mathematical Sciences, New York University,
251 Mercer Street, New York, NY 10012-1185, USA
e-mail: yap@cs.nyu.edu

of implicitly defined manifolds of higher codimension, like the solution set of a system of more than one equation in Euclidean spaces of any dimension. Like in earlier certified meshing algorithms, a crude initial sample is successively refined until a certain sampling condition is satisfied. This sampling condition guarantees that local stars of the sample points can be pasted together to form a mesh that is isotopic to the manifold. The final mesh has nice approximation properties, including good bounds on the Hausdorff distance between the manifold and the mesh, and good approximation of tangent spaces. The interesting feature is that, apart from the initialization step, the complexity of the algorithm does not depend on the dimension of the ambient Euclidean space, but only on a sampling parameter and the dimension of the manifold.

The surface–surface intersection problem often involves the conversion from an implicit representation to a parametrization of a curve. Usually, the approach consists of two steps, namely the extraction of the correct topology of the curve followed by the construction of a good approximation, usually in the form of a low-degree spline. H. Yang, B. Jüttler and L. Gonzalez-Vega present a novel method in which these steps are integrated in an evolutionary approach. An initial bounding box is shrunk to a two-dimensional implicit curve under the flow of a vector field transversal to the curve. The challenge is to certify the topology of the output, in particular by guaranteeing that all connected components of the implicitly defined curve are detected. To this end, the moving curve is represented as a B-spline curve, which may split during its evolution to adapt its topology to that of the implicit curve.

The next two papers address one of the oldest problems in mathematics: computing the complex roots of polynomials. Indeed, this is the *Fundamental Computational Problem of Algebra*, following the sobriquet of the Fundamental Theorem of Algebra (FTA). For this volume's concern with issues at the continuous–discrete interface, the Fundamental Computational Problem is pivotal—it is the primary source for the discrete geometric data (roots) that is being converted from continuous data (functions). This classical area has remained an important research topic right up to the present day. It is no surprise that such a basic problem has been studied in many forms, with many algorithms. In particular, there are two frameworks for root finding problems: on the one hand, we can split it into a prior problem of root isolation (i.e., finding regions in the complex plane that contain a unique root), followed by the problem of refining these isolated roots. On the other hand, numerical analysts often prefer to simply approximate the roots directly, without any prior isolation. This direct approximation can also lead to root isolation if root separation bounds (e.g., for integer polynomials) are available.

M. Sagraloff's paper falls under the first framework, addressing the root isolation problem for square-free real polynomials. But the formulation has a recent twist—the bits of the coefficients of the input polynomial are slowly revealed upon request. This "bit-stream model" is from a 2005 paper of Eigenwillig et al. Such a formulation turns out to be the right one for isolating the roots of a polynomial with algebraic coefficients: indeed, we do not have all the bits of the algebraic coefficients upfront, as these have to be separately computed. Important applications include solving triangular systems of multivariate polynomials and algebraic projection algorithms. Sagraloff generalizes *and* improves the complexity of recent bit-stream algorithms: (a) Previously, the bit-stream model was only applied to the Descartes method for real root isolation. In his generalization, Sagraloff can use any of the known subdivision algorithms that are based on the inclusion/exclusion predicates paradigm, for either real or complex roots. (b) The generalized framework also improves on the previous bit-stream isolation algorithms by removing random steps and improving the overall bit-complexity. The critical issue of minimizing the number of requested bits requires special computational checks, and this is inevitably related to root separation bounds. Sagraloff shows that the worst case root separation bound could now be replaced by the average root separation bounds.

The paper of P. Batra fits in the second framework of direct root approximation. He considers the homotopy path method which goes back to Weierstraß: to approximate the roots of a polynomial P_1 of degree d, begin with a nice known polynomial P_0 of the same degree and construct a homotopy $P_t (0 \leq t \leq 1)$ from the roots of P_0 to those of P_1. Weierstraß used this device to prove the FTA, but Batra turns it into a new basic constructive proof of the FTA. Beginning with approximate roots of P_0, we can generate a sequence $(\overline{x}_i)_{i \geq 0}$ of simultaneous d roots that "track" this homotopy "closely", and thus end up with approximate roots of P_1. This sequence of approximations may be generated by Newton iteration. "Tracking closely" means that each $\overline{x}_i$ is an "approximate zero of the first kind" (in the sense of Smale) for the corresponding P_{t_i}. Smale, Shub and others have extensively studied such homotopy path methods, not only for a univariate polynomial but also for systems of multivariate polynomials.

An open problem of Smale is *whether there exists a never-failing, globally convergent, iterative method for the determination of polynomial roots*. The main theorem of Batra answers this affirmatively, for the first time. Previous algorithms relied on knowing the condition number μ (distance of the path to the nearest ill-posed variety), or on probabilistic assumptions. Batra's theorem bounds the number of steps in the algorithm by $O(\mu^2)$ (the algorithm can achieve this bound without explicitly knowing μ).

In the last two decades, computational geometers have come to understand how the ubiquitous problem of non-robustness in geometric algorithms can be fully eliminated. The key problem amounts to ensuring that every branch in a program execution is taken correctly, and this reduces to determining the correct sign of computed numerical quantities. This approach is known as exact geometric computation (EGC). Although there have been successful EGC number libraries such as `leda::real` and `CORE::Expr` to solve this problem, the design space of such systems is remarkably rich and remains to be fully explored. Such an exploration is carried out by M. Mörig, I. Rössling and S. Schirra in the last paper of this issue. As the only paper in this collection with experiments as the primary focus, it only underscores the urgent need for more empirical work to match the theory, not only in EGC but at the entire continuous-discrete interface. The authors describe in detail a design of a generic `C++` number class called `Real_algebraic` that supports numbers constructed out of integers and the operations $+, -, \times, \div, \sqrt[d]{\cdot}$. A central idea in EGC number types is the necessity to construct expression DAGs which also support some kind of lazy evaluation. As usual, "generic" in `C++` means that the number class is templated; in particular the number class takes three template parameters (called Policies) that decide issues such as (1) how or when expressions DAGs are to be constructed and managed, (2) how literal (as opposed to expression) numerical quantities are represented, and (3) a data mediator between (1) and (2). A particular literal number representation explored in this work is the representation of a numerical value $\sum_{i=1}^{n} a_i$ by a sequence $a_1, \ldots, a_n$ of machine floats (two versions were implemented). Finally, the paper presents experimental results based on computing Delaunay triangulations on randomly generated point sets: the goal is to determine optimal policy choices, and also to compare the performance of `Real_algebraic` with current EGC number types in `CGAL`, `LEDA` and `CORE`.